Monotonic and Linear Relations between Growth of Quality vs Growth in Quantity in Open-Source Software Projects

[1]EKBAL RASHID ,[2]NIKOS MASTORAKIS
[1]Technical University of Sofia, Sofia, BULGARIA
[2]Sector of Electrical Engineering and Computer Science,
Hellenic Naval Academy, Piraeus, GREECE
and English Language Faculty of Engineering,
Technical University of Sofia, Sofia, BULGARIA

Abstract: - Mathematical models have been developed to study the relation between the growth of quantity and the growth of quality of open-source collaborative software repositories of GitHub. GitHub events triggering the growth of quantity and quality have been identified. Linear regression analysis, Pearson's, Spearman's, and Kendall's correlation coefficients have been used. Hypothesis testing has led to the conclusion that there may be a linear relation between quality and quantity within a certain range of values. Positive monotonic relations and dependency between quantity and quality have been strongly established. Scripts for automated testing have been developed.

Key-Words : - Software quality, linear regression models, Pearson's Spearman's Kendall's correlation coefficients, GitHub events, Software Growth, Software repositories.

Received: August 17, 2023. Revised: December 15, 2023. Accepted: February 19, 2024. Published: April 8, 2024.

1 Introduction

The noted statistician George Box had said, "All models are wrong, but some are useful." And so, when we go on to try to model the quality vs quantity factor for open-source software repositories, we are facing a myriad of issues to be tackled. The foremost being how to look at quality and quantity, how to represent them, and how to cope with the ever-changing nature of these software projects. There have been so many attempts, but hardly anyone seems to be happy. The development in the open-source world has been largely stochastic in nature, with so much randomness, that it would be very difficult to make them mathematically tractable. Over and above, the open-source world with its collaborative nature is also full of people who are learning and at the same time contributing. There are communities of developers, writers, artists, and designers, working in tandem, with many of them also involved in other day jobs, contributing to making big projects work. The projects evolve beautifully unfolding amidst their seemingly chaotic but homogeneous path of development, giving rise to extremely useful engineering products. In this paper, the authors have attempted to try and identify the events which bring about changes in these projects. GitHub is one place hosting millions of repositories. Hence, GitHub has been the point of attraction for engineers and data scientists. For each event in GitHub, a record is kept and the data of such records are available. If it is possible to identify those events that lead to quantitative changes and those that lead to qualitative changes, then data related to those events can be mined and analyzed. This is exactly what the authors have attempted to do in the present study. Parameters have been identified, normalized and bundled into two variables – quality and quantity. Two models have been suggested here. Many more can be designed likewise. One of the models suggests a strong positive linear relationship between quality and quantity. Both suggest a strong positive monotonic relationship between quality and quantity.

2 Survey of Literature

The field of mining software repositories is still an evolving area of research and hence needs to be further probed and studied. The present research is motivated by the following works. [1], has discussed about the differences between two versions of software and has shown the effect of changes in source code. This will assist the developers in understanding as to what changes in source code would cause the software to function differently. A tool called IMPEX has been discussed here. So, this is

an attempt to understand the quality of the software in terms of some quantitative change, namely the changes in source code. Again, [2], have discussed the effect of names of identifiers on the quality of the software. According to them, if the quality of names of the identifiers is of low quality, it leads to a lowering of the quality of the software as a whole. A tool has been used here to get identifier names from the source code of Java projects. They have also discussed which type of names of identifiers lead to which type of problems. Again, here the changes in the code are affecting the software quality. [3], as explained the effect of code refactoring on the quality of the software. They have developed techniques to improve the quality of the software by refactoring code. The authors have tried to formalize the method of refactoring code in this paper. [4], have in a similar manner analyzedanalyzed structural changes in software versions and the chief motive was to measure changes related to the structure of source code in the software. Although there is no attempt to relate the changes in code structure with the quality of the software, it is quite evident that there may be a relation between the two and this may constitute an interesting study. However, these works are looking into the software projects only from a static point of view. It is necessary to look at these projects from a dynamic perspective, from the viewpoint of their ever-changing nature, from the viewpoint of their coming into being and going out of being. This has been the primary focus of the present paper.

Several studies are there that have tried to understand features of open-source projects and parameters such as the number of active contributors, the use of different kinds of programming languages, the particular structure of the project, and on many other important parameters that the author considers to be quantitative parameters. Many such works are found in [5] and [6]. Again, these are looking largely at popularity. There has to be a judgement of quality too using suitable metrics based on events of open-source projects.

There are other works such as [7] and [8] that analyze the data in datasets related to the GitHub repositories. In such studies, mainly the stars, forks, and issues are considered. Many have also included code and outline of how the data has been retrieved, that is they have elaborated in detail the mining methods involved. The study also did a random word selection from a certain word list which was given to a GitHub API and then the API gave back a list of repos. Out of the list, some are selected at random and mining is done to extract the data of those reports.

It has used the metric of popularity as $popularity = stars + forks + pulls^2$. The authors in this study have tried to correlate the documentation of the project to this defined value of popularity. The method though is not discussed in great detail], but it does motivate the present study to think in similar terms.

In [9], authors have adequately described that a large number of GitHub repositories are personal and not active. This may have a large effect on the conclusions that one may draw from a dataset of GitHub repositories. For this, the authors analysed parts of GHTorrent datasets and sent surveys to users of GitHub. They also highlighted the fact that there was a substantial number of projects that had very few commits so it might not be proper to jump to conclusions from the commit data of GitHub.

In [10], authors have shown that frequency of commits and the evolution of versions of files in eight large projects of GitHub have a certain degree of correlation. The projects discussed here are very successful. It presented a picture of the number of commits and the number of lines of code being changed in each file and a comparison between the number of commits and file changes in different versions etc. All these works are pointing towards the attempt to design a software quality vs quantity model for understanding the relation between them.

3 Methodology

Detailed mining has been done from a public dataset available on Google Big Query. The total data processed in Big Query went to about 43.2 TB. More than 170 queries were performed on the dataset to extract the data. Since the process is cost-consuming, it could be performed for only a single time. This extracted data was cross-checked from Click House [11] and all the data was tabulated. It needs to be mentioned here that GH Archive has made available the data from GitHub for the last eleven years, that is, from 2011 to 2021. This has a detailed collection of several events. The events and their identifiers are as below:

1. CommitCommentEvent: triggered when there is a comment in a commit
2. CreateEvent: when there is the creation of a branch or a tag
3. DeleteEvent: When a branch or a tag is deleted

4. ForkEvent: This is triggered when a user forks any repository
5. GollumEvent: When a wiki page is created or updated
6. IssueCommentEvent: When there is a comment on any issue opened
7. IssuesEvent: This event is related to an issue. There are many actions involved with it such as opened, closed, reopened, assigned, etc.
8. MemberEvent: This is an event related to the activity of any member of the repository
9. PublicEvent: When a repository is made a public repository
10. PullRequestEvent: This event is triggered when a pull request is made and this also includes many actions like opened, closed, reopened, assigned, etc.
11. PullRequestReviewEvent: Whenever there is a review about a pull request.
12. PullRequestReviewCommentEvent: If anyone comments on a pull request review
13. PushEvent: Whenever one or more commits are pushed to a repository branch
14. ReleaseEvent: Whenever a new version is release, this event is triggered
15. SponsorshipEvent: Event associated with the listing of sponsorship
16. WatchEvent: When anyone stars a repository

For this study, some GitHub projects have been randomly selected based on popularity. Three types of popularity measures have been considered as follows (Table 1):

i) Highly rated projects: Projects with stars greater or equal to 50,000
ii) Moderately rated projects: Projects with stars between 5,000 and 10,000
iii) Slightly rated projects: Projects with stars less or equal to 1,000

The above repositories have been arbitrarily selected by running queries as per GitHub docs, [12], and then listing them out. Only one criterion has been considered while selecting the repositories, that is, the repositories should hold coded projects. There are collections of images, or books, or other resources as GitHub repositories, but they have not been considered for this study.

The raw data was then processed to properly ascertain the quantitative, qualitative, and popularity parameters. Although there may be many ways in which growth in quantity, growth of quality, and popularity can be defined for open-source collaborative projects like GitHub, this research has identified the following at present:

Parameters measuring growth in quantity:

a) Pull Request Activity: It is the ratio between the number of pull requests opened and the number of pull requests closed. However, here the ratio itself is not taken as the quantitative measure. Rather, it is normalized. The reason for this is that different types of repositories are taken up for data collection. A less active repository may have the same value of closed pull requests to opened pull requests ratio with a far lesser number of pull requests. Hence, this ratio has been normalized by multiplying it by a factor proportional to the closed number of pull requests. The final normalized pull request activity is as follows:

$$\text{pull request activity} = \frac{no.of\ closed\ pull\ requests}{no.of\ open\ pull\ requests} \text{ x } \frac{no.of\ closed\ pull\ requests}{100}$$

where (closed pull requests/100) is the normalizing factor. There would be not much difference in the final results had we simply multiplied the original ratio with the number of closed requests without dividing it by 100, but it has been done to keep the values small. This normalized pull request activity has been calculated for all the listed projects for a period of seven years (2015-2021).

b) Issue Activity: It is the ratio between the number of closed issues to the number of issues opened. Similar to the above parameter, there is again a need for normalizing the data and for that the following method has been adopted:

$$\text{issue activity} = \frac{no.of\ closed\ issues}{no.of\ open\ issues} \text{ x } \frac{no.of\ closed\ issues}{100}$$

As stated above, the value 100 in the denominator of the normalizing factor does not matter much so far as the final results are concerned. The normalized issue activity has been calculated for all the listed projects for a period of seven years (2015-2021).

After calculating the pull request activity and the issue activity, the growth of quantity has been determined as:

growth in quantity = normalized pull request activity + normalized issue activity

1. Parameters measuring growth of quality:

Open-source projects follow a different quality assurance model and quality control processes. The pivotal point in these methods is peer-review of code and local testing before updating the repositories.

Hence, any update in the form of pushes or the release of new versions signals extensive implementation of quality standards. For this reason, the author has taken these two events as quality parameters.

a) Number of pushes: A push will update the remote repository branch with a commit. When a code has been written to serve as a patch or for some other feature, it is reviewed and tested locally. Only after adequate alpha testing activities, it is pushed to the remote branch. Hence, a push will improve the quality of the project as it is reviewed and tested code. However, a single push may not affect the quality to a great extent. Hence to use it as a parameter, each push has been treated to raise the quality of the project by a factor of 0.1. The number of pushes for the projects listed above has been extracted for seven years (2015 to 2021) and tabulated.
b) Number of releases: The release of a version cannot be a qualitative leap in the development of collaborative open-source projects. The pushes to the rawhide branch are effectively quality changers, but they are in the larger perspective small quantitative changes in quality that are more or less imperceptible to the common user. However, the release of a new version brings about a sudden leap, a visible qualitative change in the software which is revolutionary in character and which has a break from the earlier version of the project. This change in quality is much more pronounced than the qualitative change brought about by the push event. Hence the normalized value of this parameter is obtained by multiplying the number of releases by unity, signifying that the quality has grown ten times compared to a push event. Of course, this may be a matter of discussion whether it would be proper to consider the growth of quality in this fashion, or what should be the value multiplied to get a suitable normalized quality growth parameter.

The data for seven years (2015-2021) have been calculated for all the projects listed above. After this has been done the value showing the growth of quality is calculated for each year and each project using the following formula:

growth in quality = (no. of pushes)/10 + (no. of releases)

Where the term (no. of pushes)/10 is the normalized number of pushes.

'Growth in quantity' has been taken as the independent variable (*x)* while the 'growth of quality' has been taken as the dependent variable *(y).*

Pearson's correlation coefficients, [13], Spearman's correlation coefficients, and Kendall's correlation coefficients have been evaluated for understanding the relation between them. The detailed methods to find out these coefficients have been explained in several noted works, [13], [14], [15], [16], which have been consulted meticulously. The details have been skipped with the assumption that the reader can easily access these works or any other related literature.

Several models have been framed with different upper and lower limits of parameters. The most suitable model has been suggested for drawing inferences. 80% of the data has been used for training the models and 20% data has been used for testing the models. Hypothesis testing has been done by calculating probability values with a significance level of 95%. Finally, a regression analysis has been done to estimate the mathematical relation between the growth in quantity and the growth of quality. For this entire activity, an entire Python script has been developed for automated testing. The script has been executed using Jupyter. It takes values from the Excel sheets of different models and generates graphs and necessary analytical material for drawing inferences.

We know from the principles of Hypothesis testing as elucidated in many noted works such as [17], [18], [19], [20] that the null hypothesis is rejected if the P-value is less than 0.05 and this is the method used here to test the hypothesis. The hypothesis is as follows:

H_0: The correlation coefficient is not significantly different from zero. There does not exist a linear relation between growth in quantity and growth of quality in the population.

H_a: The population correlation coefficient is significantly different from zero. There is a significant linear relationship between growth in quantity and growth of quality in the population.

Table 1. Three types of popularity measures

	highly rated projects		moderately rated projects		slightly rated projects
	greater or equal to 50,000 stars		between 5,000 and 10,000 starts		less or equal to 1,000 stars
1	vuejs/vue	11	knockout/knockout	21	marcelstoer/nodemcu-pyflasher
2	facebook/react	12	cyclejs/cyclejs	22	sebleier/django-redis-cache
3	twbs/bootstrap	13	jquery/jquery-mobile	23	microsoft/coyote
4	flutter/flutter	14	code4craft/webmagic	24	liberodark/Odrive
5	microsoft/vscode	15	nasa/openmct	25	data-forge/data-forge-ts
6	tensorflow/tensorflow	16	ansible/awx	26	Olivine-Labs/busted
7	facebook/react-native	17	brianc/node-postgres	27	MetalPetal/MetalPetal
8	electron/electron	18	openresty/openresty	28	cbeuw/Cloak
9	nodejs/node	19	appwrite/appwrite	29	arnesson/cordova-plugin-firebase
10	angular/angular	20	teamcapybara/capybara	30	hahnlee/hwp.js

4 Mathematical Models

Model 0: In this model, the entire data set has been considered without any edits. This contains data in two columns - namely quality and quantity for the selected thirty repositories over a period of seven years. For some of the repositories, the data for some particular years was not available. In those cases, the value zero has been used. So, a zero in this model stands for the non-availability of data. The Python scripts generate the following figures and regression data for this model and it can be seen in Figure 1, Figure 2, Figure 3, Figure 4, Figure 5, Figure 6, Figure 7, Figure 8, Figure 9, Figure 10, Figure 11, Figure 12, Figure 13, Figure 14, Figure 15 and in Table 2 and Table 3.

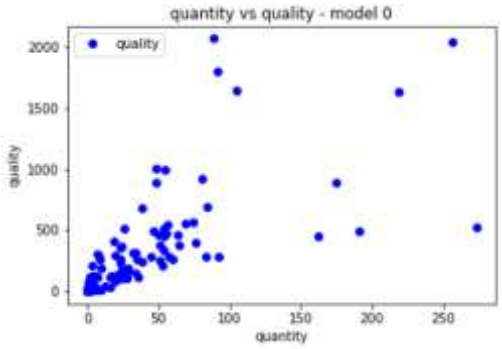

Fig. 1: Quantity vs Quality

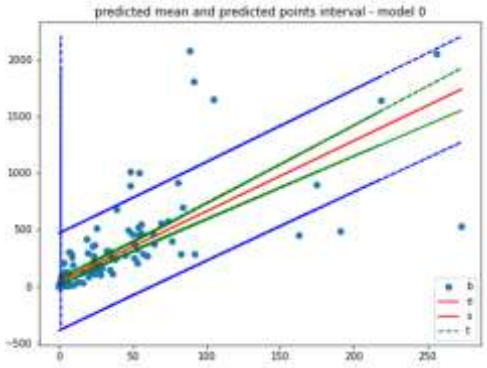

Fig. 2: Predicted mean and predicted points interval

Table 2. Regression summary (model0) without constant - OLS Regression Results

regression summary (model0) without constant - OLS Regression Results

Dep. Variable:	quality	R-squared (uncentered):	0.648
Model:	OLS	Adj. R-squared (uncentered):	0.647
Method:	Least Squares	F-statistic:	385.2
No. Observations:	210	Prob (F-statistic):	2.60e-49
Df Residuals:	209	Log-Likelihood:	-1429.5

	coef	std err	t	P>|t|	[0.025	0.975]
quantity	6.5838	0.335	19.626	0.000	5.922	7.245

Table 3. Regression summary (model0) with constant - OLS Regression Results

regression summary (model0) with constant - OLS Regression Results

Dep. Variable:	quality	R-squared:	0.580
Model:	OLS	Adj. R-squared:	0.578
Method:	Least Squares	F-statistic:	287.7
No. Observations:	210	Prob (F-statistic):	4.35e-41
Df Residuals:	208	Log-Likelihood:	-1426.5

	coef	std err	t	P>|t|	[0.025	0.975]
const	40.7692	16.498	2.471	0.014	8.244	73.294
quantity	6.2020	0.366	16.961	0.000	5.481	6.923

train data y intercept = 50.064551139263427
train data r-squared value = 0.5353810007950
mean squared error = 9865.823488859845
test data r-squared value = 0.3651989138067

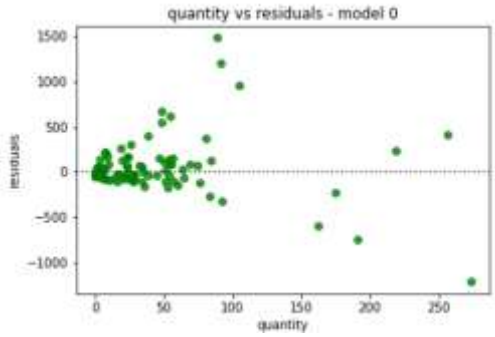

Fig. 3: Quantity vs residuals

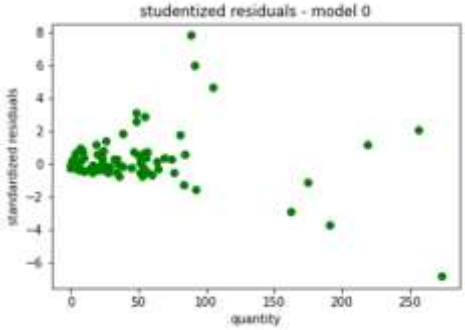

Fig. 4: studentized residuals

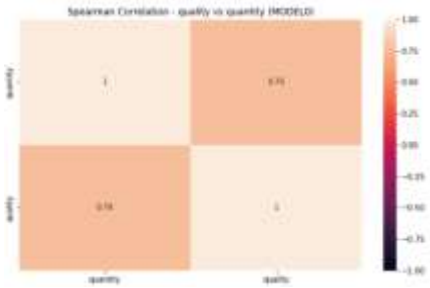

Fig. 5: Spearman correlation

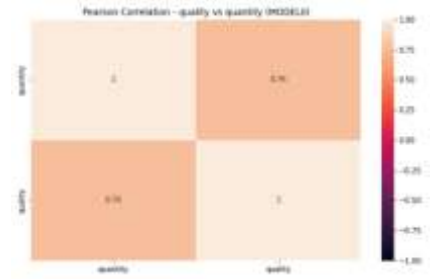

Fig. 6: Pearson correlation

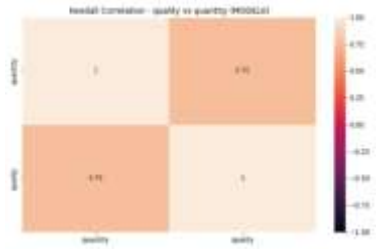

Fig. 7: Kendall correlation

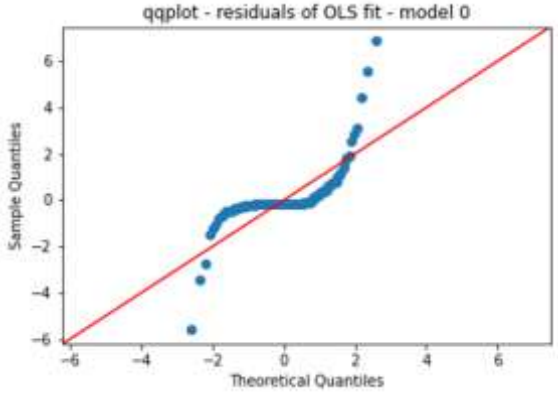

Fig. 8: qqplot residuals

The correlation coefficient without constant is fairly high. The correlation coefficient with the coefficient is also sufficiently high. In both cases, the P-values are way below the 0.05 value. That may lure the statistician to reject the null hypothesis. However, the residuals of the OLS fit do not satisfy the condition of being normally distributed. Besides the studentized residuals also do not fall within the range from +2 to -2. We therefore cannot accept the regression model. We may not conclude anything about the linear relationship between quality and quantity based on this model. This is probably because of the outliers existing in the model. We can see from the graph of predicted mean and predicted point intervals that many points are lying outside the interval. The mean squared error is also quite high. Hence, so far as the question of linear regression is considered, we have to reject this model. However, Spearman's coefficient is quite valid and it suggests a strong positive monotonic relationship between quality and quantity. Kendall's coefficient also does not rely on any assumption and it has a high value suggesting that there is a strong dependence of quality on quantity. Therefore, although this model does not succeed in terms of establishing a linear relationship, it does suggest strong dependence and strong positive monotonic relationship between quantity and quality.

Model 1: In this model, an attempt has been made to remove outliers. For this quality values which are less than 13 have been removed. The selection of this value 13 is arbitrary and it has been done only to supervise the reduction of outliers. It has been assumed that values below 13 may not fit the model. Similarly, quality values above 600 have been deleted, assuming that pure collaborative work may result in some upper limit of quality. Projects achieving higher quality values may be artificially achieved by using the non-random involvement of developers. Hence there are no zero values for this model. The Python script generates the following regression analysis See Table 4 and Table 5:

Table 4. Regression summary (model1) without constant - OLS Regression Results

regression summary (model1) without constant - OLS Regression Results

Dep. Variable:	quality	R-squared (uncentered):	0.798
Model:	OLS	Adj. R-squared (uncentered):	0.796
Method:	Least Squares	F-statistic:	347.3
No. Observations:	89	Prob (F-statistic):	2.67e-32
Df Residuals:	88	Log-Likelihood:	-544.38

	coef	std err	t	P>|t|	[0.025	0.975]
quantity	6.6444	0.357	18.636	0.000	5.936	7.353

Table 5. Regression summary (model1) with constant - OLS Regression Results

regression summary (model1) with constant - OLS Regression Results

Dep. Variable:	quality	R-squared:	0.608
Model:	OLS	Adj. R-squared:	0.603
Method:	Least Squares	F-statistic:	134.8
No. Observations:	89	Prob (F-statistic):	2.25e-19
Df Residuals:	87	Log-Likelihood:	-536.87

	coef	std err	t	P>|t|	[0.025	0.975]
const	60.5005	15.122	4.001	0.000	30.443	90.558
quantity	5.3543	0.461	11.612	0.000	4.438	6.271

train data y intercept = 71.96519057805702 mean squared error = 12726.304444199732
train data r-squared value = 0.596280969212 test data r-squared value = 0.6212809762850

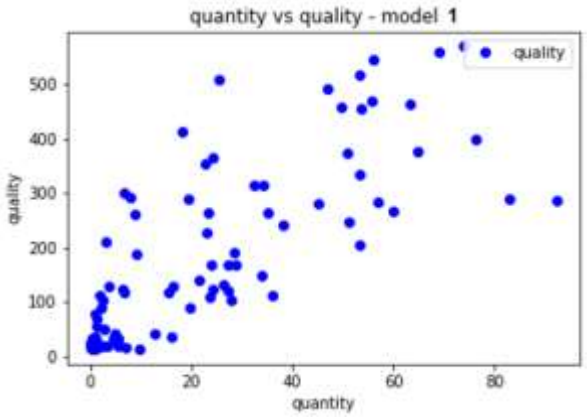

Fig. 9: Quantity vs Quality for model 1

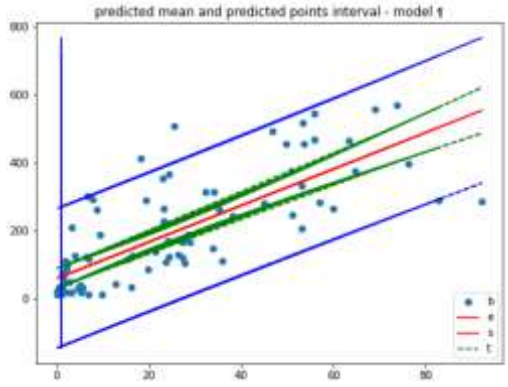

Fig. 10: predicted mean and predicted points interval for model 1

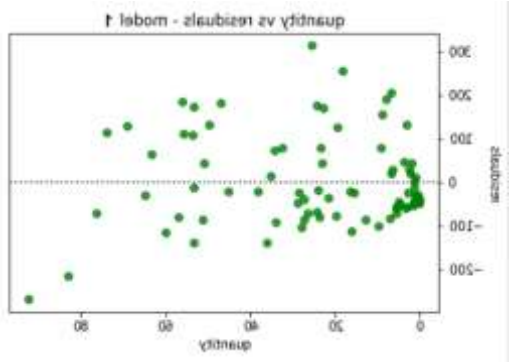

Fig. 11: Quality vs residuals for model 1

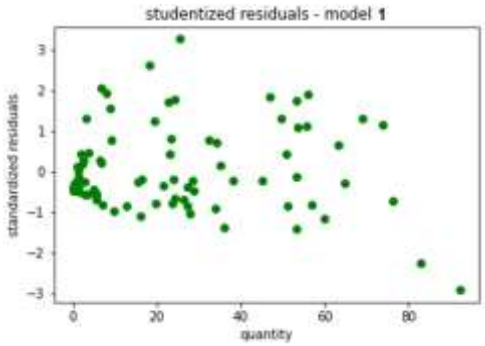

Fig. 12: studentized residuals for model 1

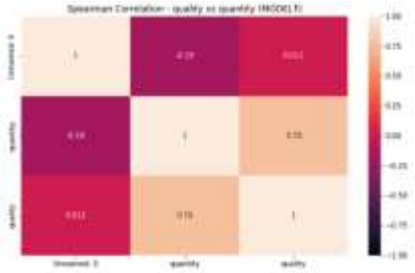

Fig. 13: qqplot residualsof OLS fit for model 1

Fig. 14: Spearman correlation for model 1

Fig. 15: Pearson correlation for model 1

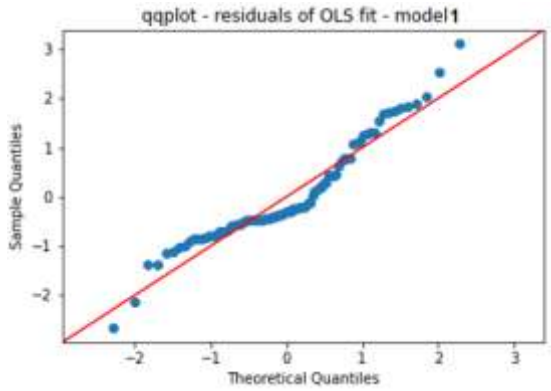

Fig. 16: Kendall correlation for model 1

The correlation coefficient without constant is fairly high. The correlation coefficient with the coefficient is also sufficiently high. In both cases, the P-values are way below the 0.05 value. Almost all the residuals fall within the range from +2 to -2. Almost all the residuals are along the 45-degree line and thus, normally distributed. Hence, the assumptions for the linear regression model are satisfied. We can accept this model. Since the P-values are far below 0.05 and the correlation coefficient is sufficiently high, we can reject the null hypothesis based on this model. We can say that a linear correlation exists between quantity and quality for this model. The relation between quality and quantity can be expressed using the equation:

$$y = 5.3543\ x + 60.5005$$

Suggesting that for every unit quantitative increase, the quality increases by a factor of 5.3543.

Moreover, Spearman's coefficient is quite valid and it suggests a strong positive monotonic relationship between quality and quantity. Kendall's coefficient also does not rely on any assumption and it has a high value suggesting that there is a strong dependence between quality and quantity. Based on this model, we may say that within the given range of quality values selected, there exists a strong positive monotonic and linear relationship between growth in quantity and growth of quality of collaborative software projects.

The predicted mean interval shown in the graph of this model is also very encouraging. It suggests that the mean quality may be maintained in this interval if the quantitative changes are monitored accordingly.

5 Conclusions

The relation between the growth of quality vs growth in quantity can be modeled in the manner demonstrated above. For both the models shown above, we may safely conclude based on Kendall's coefficient that there exists a dependence between quantity and quality where the terms quantity and quality are as defined in this paper. Similarly, we may also conclude that there exists a strong positive monotonic relationship between growth of quantity and growth of quality on the basis of Spearman's coefficient values as seen in both models. Both Spearman's and Kendall's coefficients do not rely on any assumptions and can be used for drawing these conclusions. However, for the first model, the regression analysis does not meet the test assumptions and hence cannot be used for determining linear relationships. The second model has passed the test assumptions of a linear regression model. Hence, we can use it. The correlation coefficient in this model is significantly high. The P-values are much lower than 0.05. This is true for both cases – the OLS regression without a constant and with a constant. Due to this, we can reject the null hypothesis and accept the alternate hypothesis. We therefore conclude based on the second model that there exists a strong positive linear relationship between growth in quantity and growth of quality of collaborative software projects within a suitable range of values as presented by the model.

6 Significance

The mathematical models suggested in this paper may serve as the basis for future research to study the relationship between quality and quantity in further detail. A relationship between quality and quantity may suggest that the quality of the software project may be estimated or even predicted by measuring the growth in quantity. This may lead to the development of newer software development models so far as the open-source collaborative software world is concerned. There may be situations where the direction of development may be controlled based on such quality vs quantity mathematical models. Supervised learning of such types also may give us an insight into the nature of the changing world of software projects. A study about how stochastic development takes place in the realm of software projects of such types may also be initiated.

7 Future Scope

Quantity and quality have been defined based on four GitHub events, namely – pullRequest, issue, push, and release. However, there are many other events in the lifecycle of open-source collaborative projects. Interpretation of such other events may lead to more interesting results. This paper has discussed two important models. The objective has been to demonstrate the technique of mathematical modeling in this field. Further situations may be modeled and used to arrive at conclusions. A particular range of data has been used to model a situation. Other ranges of data may be considered and more models can be developed. The y-intercept and slopes may be interpreted in other ways too.

References:

[1] D. Nemer. IMPEX: An Approach to Analyse Source Code Changes on Software Run Behaviour. *Journal of Software Engineering and Applications*, 2013, vol. 6, no 4.

[2] S. Butler, M. Wermelinger, Y. Yu and H. Sharp. Exploring the Influence of Identifier Names on Code Quality: An Empirical Study. *14th European Conference on Software maintenance and Reengineering*, Madrid, Spain, 2010.

[3] B. D. Bois, T. Mens. Describing the impact of refractoring on internal program quality. *ELISA workshop*, Amstredam, Netherlands, 2003.

[4] C. Gerlec, M. Hericko. Analysing Structural Software Changes: A Case Study. *BCI-LOCAL 2012*, Novi Sad, Serbia, 2012.

[5] Aggarwal K., Hindle A., Stroulia E. Co-evolution of project documentation and popularity within GitHub in *Proceedings of MSR 2014, ACM*, pp.361-362. ISBN: 978-1-4503-2863-0.

[6] T. F. Bisisyande. Got Issues? Who cares about it? A large-scale investigation of issue trackers from GitHub. *IEEE 21st international Symposium on Software Reliability Engineering (ISSRE)*. Nov. 2013 pp. 189-196.

[7] Jarczyk O. and others in GitHub Projects. Quality Analysis of Open-Source Software. *Social Informatics Springer*, Cham, Nov. 2014 pp. 81-89.

[8] Peterson Kevin on Mining GitHub: Why Commit Stops Exploring the Relationship between Developer's Commit Pattern and File Version Evolution. *20th Asia Pacific Software Engineering Conference*, Vol. 2, Dec 2013, pp. 164-170.

[9] Kalliamvakou E. and others. The Promises and Perils of Mining GitHub. *Proceedings of the 11th Working Conference on Mining Software Repositories MSR 2014 ACM*, pp. 92-101.

[10] Weicheng Y., Beijun S., Ben X. Mining GitHub: Why Commit Stops Exploring the Relationship between Developer's Commit Pattern and File Version Evolution. *20th Asia Pacific Software Engineering Conference*, Vol. 2 Dec. 2013.

[11] GitHub: Everything You Always Wanted to Know about GitHub (But Were Afraid to Ask), [Online]. https://gh.clickhouse.tech/explorer/ (Accessed Date: April 30, 2022).

[12] GitHub Docs, [Online]. https://docs.github.com/en/github (Accessed Date: April 30, 2022).

[13] Pearson Karl Mathematical Contributions to the Theory of Evolution. V. *On the Reconstruction of the Stature of Prehistoric Races. London: Philosophical Transactions of the Royal Society of London*, 1898. First Edition.

[14] Chen P. Y., Popovich P.M., *Correlation, Parametric and Non-Parametric Measures*. A Sage University Publication, 2002. --14

[15] Bobko P., *Correlation and Regression Applications for Industrial Organizational*

Psychology and Management, Sage Publication, New Delhi, Second Edition, pp. 67-84.--15

[16] Gupta S.P., Gupta M.P. *Business Statistics Sultan Chand & Sons*, 2001, Twelfth Edition, pp. 237-247

[17] Hartshorn S., Hypothesis Testing: A Visual Introduction to Statistical Significance, Kindle Edition, Amazon, [Online]. https://www.amazon.in/Hypothesis-Testing-Introduction-Statistical-Significance-ebook/dp/B019N212NE (Accessed Date: April 30, 2022).

[18] Erich L. Lehmann, Joseph P. Romano, *Testing Statistical Hypotheses*, Springer-Verlag New York, ISBN: 978-0-387-27605-2, pp. 56-78.

[19] Arthur Taff. 2018. *Hypothesis Testing: The Ultimate Beginner's Guide to Statistical Significance. CreateSpace Independent Publishing Platform*, North Charleston, SC, USA. pp. 69-75.

[20] Frost Jim, *Hypothesis Testing: An Intuitive Guide for Making Data Driven Decisions*, Kindle Edition, March 2021, ISBN: 978-1735431154.

Contribution of Individual Authors to the Creation of a Scientific Article (Ghostwriting Policy)

The authors equally contributed in the present research, at all stages from the formulation of the problem to the final findings and solution.

Sources of Funding for Research Presented in a Scientific Article or Scientific Article Itself

No funding was received for conducting this study.

Conflict of Interest

The authors have no conflicts of interest to declare that are relevant to the content of this article.